Mobies Ethereal Sting OEP
The Ptolemy II Experiment

Edward A. Lee
Professor
UC Berkeley

E0 Implementation in Ptolemy II

Authors:
Mark Oliver (WPAFB)
Steve Neuendorffer
Edward Lee

This model shows the SWRI baud-rate detection algorithm implemented in Ptolemy II.
Automatic code generation enables rapid implementation from high-level component-based design.

We are developing a code generation technique based on component specialization that transforms Ptolemy II models into a Java system implementation.

From Model to Implementation

- **Generator-based code generation**
  - Done in Ptolemy Classic
  - Library maintenance is very expensive

- **Native Java compiler**
  - Drags in the development environment
  - Result is large, and has unpredictable timing

- **Component specialization**
  - Produce minimized Java implementation
  - Minimize or eliminate dynamic memory management
  - Compile to the target platform using one of:
    - Java to C translation
    - Native Java compiler
    - Just-in-time compiler
    - Native Java platform (e.g. Dallas Tini boards)
Component Specialization

Model of Computation semantics defines communication, flow of control

Ptolemy II model

Java actor definitions are parsed and then specialized for their context.

abstract syntax tree

Schedule:
- fire Gaussian0
- fire Ramp1
- fire Sine2
- fire AddSubtract5
- fire SequenceScope10

Limitations Exposed by the Experiment

- No actor for array maximum
  - Added later by Mark Oliver, built into library
  - Easy workaround used very wide signal busses
- Type resolution was very slow when using very wide signal busses
  - Fixed by Steve Neuendorffer
- AudioReader actor was unfinished
  - Didn’t use FileAttribute
  - Didn’t correctly deliver stereo signals
- FFT actor performs only radix-2 FFTs
  - Could use MATLAB interface to generalize
- Component specialization framework limitations
  - Didn’t handle FileAttributes
  - Error handling the absolute() function
  - Error specializing AudioReader
Log of Effort

- Three active participants, plus some spectators:
  - 0.5 hours examining EtherealSting website and figuring out what to do.
  - 2 hours constructing and experimenting with the model to detect the baud rate. This was built by modifying a model constructed by Edward Lee at the Mobies PI meeting (which took, perhaps, 1.5 hours to build).
  - 1 hour fixing bug in AudioReader actor to use FileAttribute.
  - 4 hours experimenting with component specialization.
  - Total time: 9 hours
- 6.5 hours fixing bugs exposed by the experiment.
- The experiment stimulated further work on comm/signal processing libraries.

Actor Libraries - Signal Processing

Capabilities:
- filtering
  - multirate polyphase FIR, IIR, lattice, LMS adaptive filter, dot product, up/downsample
- random numbers/signals
  - Bernoulli, Gaussian, Rician, Rayleigh, Uniform, arbitrary discrete distributions
- linear system generators
- spectral estimation library
  - FFT, periodogram, maximum entropy
- comm functions:
  - Viterbi decoder (MLSE), convolutional/black coder/decoders, PN sequence generation, scrambling/descrambling, raised cosine
- array and matrix operations
- rich expression language / actor
  - extensive function library
  - MATLAB-like matrix comprehension
  - higher-order functional semantics
  - sophisticated, integrated type system
- interpolator, phase unwrap, lookup table, signal generators, trig functions
- signal plotters
- extensive image processing library
  - based on Java JAI, JMF
- audio interfaces
Supervisory Structure
Experimental SA Compute Resource

Model-based compute resource:

PushConsumer actor receives pushed data provided via CORBA, where the data is an XML model of an SA algorithm.

MobileModel actor accepts a StringToken containing an XML description of a model. It then executes that model on a stream of input data.

Supervisory Structure
Experimental Task Manager

Model-based task manager:

PushConsumer actor receives pushed data provided via CORBA, where the data is a user request for signal analysis.

PushSupplier sends an XML representation of an SA model via CORBA.

Supervisor state machine has resource allocation logic.

Authors:
Yang Zhao
Steve Neuendorffer
Xiaojun Liu

Lee, U. C. Berkeley 9

Lee, U. C. Berkeley 10
Supervisory Structure
Experimental User Model

User model:

PullSupplier actor provides signal data on demand from SA algorithm

PushSupplier actor sends a request for signal analysis to the task manager.

To Do

- Handle failures of mobile model
  - use "model error handler" mechanism in Ptolemy II
- Secure execution of mobile model
  - all Java code executed is locally defined
  - mark actors and directors that convey no authority
  - set MobileModel security level to restrict actors
- Encrypted communication of models & data
  - currently XML plain text
- Authenticated access to MobileModels
  - consider using "capability" mechanisms
  - use peer-to-peer technology to "discover" capabilities.
Another Application: Controlling the Caltech Ducted Fan Vehicle

This effort is applying Mobies technology to the SEC program.

Caltech Vehicles

Difficulties:
1) Complex control problem
2) Complex implementation platform

Command computer: Waypoints, trajectories, Control changes

Localization computer estimates vehicle locations

Vehicles with onboard controllers and 802.11b
A Detailed Heterogenous Model

**Measured Physical Parameters**

- Mass: 5.15
- Moment: 0.5
- psi: 0.04
- eta: 5.5
- r: 0.124
- initial_omega: 0
- initial_x_velocity: 0
- initial_y_velocity: 0

**Discrete Event model convenient for events that do not occur at the same time.**

**Model of computation and communication delay.**

Array of 3 Bytes:
{85, Left, Right}
Sent immediately after controller computes value

Array of 50 Bytes:
{TimeStamp, ID, X, Y, Angle}
60 times a second

**Continuous time model of vehicle dynamics**

**Fan Thrust Map**

**Data formatting**
A Detailed Heterogenous Model

Discrete-state model of vehicle software

Encapsulated Control Law

Towards Implementation

802.11b

RS-232
**Hardware-in-the-loop**

Replace hardware-true simulation model with actual vehicle.

Allows validation of continuous dynamics model, and hardware/software interface.

**Simulation-in-the-loop**

Code generation of the controller onto an embedded platform.

Allows validation of generated code, and execution delay.
System Implementation

The generated code forms the final system implementation.

Controller Updates

Simplified model of base station

Controller component transmitted over publish/subscribe network

Mobile model allows substitution of different controllers

Authors:
Steve Neuendorffer
Yang Zhao

Authors:
Crazyboard
MobileModel
TimedDelay
PushConsumer
PushSupplier
FileReader
SingleEvent
DE Director